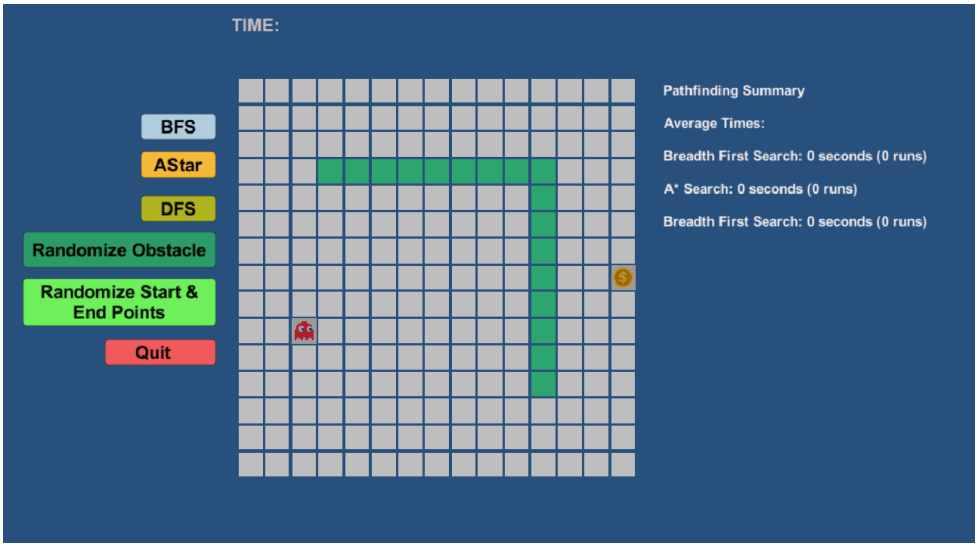
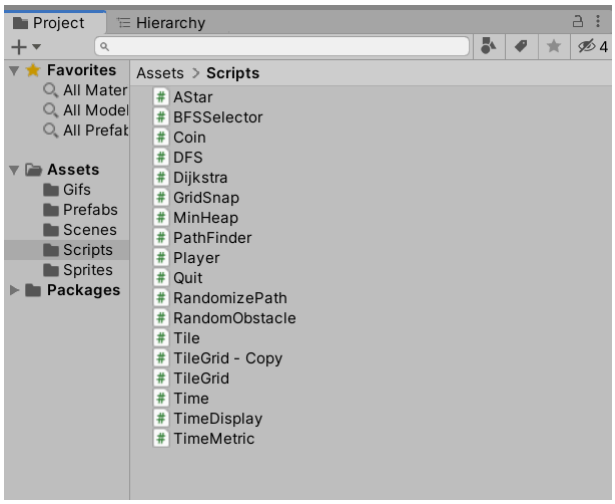
Summary:

In this activity, students have the opportunity to explore how various search algorithms (e.g., breadth-first search, depth-first search, A*) enable NPCs to navigate from a starting point to a desired endpoint within a game level that features a grid space with assorted obstacles.



Above is a screenshot of the activity. The character and coin represent the start and end endpoints. The green tiles are the obstacles. On the right are the buttons students can select to see the display of each algorithm from the start to the end point bypassing the obstacles. The students also have the options to randomize the points and obstacles to see how each algorithm works in different game environments. A new addition is the time metrics. On the top students can see how long it took for each algorithm to run. Under the pathfinding summary, the activity keeps track of the average times it has taken the algorithm run and how many runs of each algorithm the student has done.

Scripts:



TileGrid.cs - Manages the grid environment and the display of each algorithm when the buttons are clicked. Also has the logic for the randomized obstacles & start/end points as well as the display of the time metrics on the side (pathfinding summary).

Tile.cs - Creates the tiles of the grids

PathFinder.cs, MinHeap.cs - Manages the logic for each search algorithm, minimum heap, and how the tile colors change based on which tiles each algorithm is searching at. This is where you can change the character displays on the tiles.

AStar.cs, BFSSelector.cs, Dijkstra.cs, RandomizePath.cs, RandomObstacle.cs, Quit.cs - These manage whether the button was clicked or not by the user.
TimeDisplay.cs - Manages the logic to calculate the start and end time of each algorithm. When each algorithm button is clicked this class will be called and basically mirrors a timer with the start time and end time to calculate the average time. This also displays the time at the top of the activity.

*Any other files in the scripts folder shouldn't be necessary but can be used as reference for any future implementations.

Other Notes:

- The characters used are different prefabs and keep be adjusted or changed in the prefab folder
- One thing I tried changing was the speed of the algorithms displaying. The speed can be changed in the PathFinder.cs file but it was going either way to fast or way to slow
    - This is something that is still being looked at to improve

Future Goals:

- Make the game more interactive by allowing users to select their own start & end points as well as obstacles (ex. Drag and drop tiles or select tiles)
- Log the data collected
- Generate different game levels and complexity to see search algorithms in different game environments